4

Komentari

„Ne komentarišite loš kod – napišite ga ponovo.“
– Brajan V. Kernighan i P. J. Plaugher[1]

Ništa ne može biti toliko korisno kao dobro ubačen komentar. Ništa ne može zagušti modul više od neozbiljnih priča u prazno u komentarima. Ništa ne može biti toliko štetno kao stari surov komentar koji propagira laž i dezinformacije.

Komentari nisu poput Šindlerove liste. Oni nisu „nasušna potreba.“ Zaista, u najboljem slučaju komentari su nužno zlo. Da su naši programski jezici dovoljno izražajni i da imamo talenat da suptilno koristimo te jezike da bismo izrazili svoje namere, komentari nam ne bi bili toliko potrebni, a možda bismo mogli i bez njih.

1. [KP78], p. 144.

Pravilna upotreba komentara je kompenzacija za naš neuspeh da se izrazimo u kodu. Obratite pažnju da sam upotrebio reč reč *neuspeh*. To sam i mislio. Komentari su uvek nedostaci. Moramo da ih imamo jer ne možemo uvek da smislimo kako da se izrazimo bez njih, ali njihova upotreba nije razlog za slavlje.

Dakle, kada se nađete u položaju da morate da napišete komentar, razmislite o tome i vidite da li postoji način da se izrazite kroz kod. Svaki put kada se izrazite kodom, čestitajte sebi. Svaki put kada napišete komentar, napravite grimasu i budite svesni svoje nesposobnosti izražavanja.

Zašto sam takav po pitanju komentara? Zato što lažu. Ne uvek i ne s namerom, ali se to događa previše često. Što je komentar stariji, dalje je od koda koji opisuje i verovatno je pogrešan. Razlog je jednostavan. Programeri ne mogu da održavaju komentare ažurnim.

Kod se menja i razvija. Delovi koda se premaštaju tamo-ovamo. Ti delovi koda se dele i razmnožavaju i ponovo okupljaju u grupama. Nažalost, komentari ih ne prate uvek, štaviše *ne mogu* uvek da ih prate. Previše često se komentari odvajaju od koda koji opisuju i postaju siročad sve manje tačnosti. Na primer, pogledajte šta se dogodilo sa ovim komentarom i redom koji je trebalo da opiše:

```
MockRequest request;

private final String HTTP_DATE_REGEXP =
"[SMTWF][a-z]{2}\\,\\s[0-9]{2}\\s[JFMASOND][a-z]{2}\\s"+
"[0-9]{4}\\s[0-9]{2}\\:[0-9]{2}\\:[0-9]{2}\\sGMT"; private Response response;
private FitNesseContext context;
private FileResponder responder;
private Locale saveLocale;
// Example: "Tue, 02 Apr 2003 22:18:49 GMT"
```

Ostale instance promenljive koje su verovatno dodate kasnije, ubačene su između konstante `HTTP_DATE_REGEXP` i pratećeg komentara.

Moguće je reći da programeri moraju da budu disciplinovani i da treba da održavaju komentare na visokom stepenu ažurnosti, relevantnosti i tačnosti. Slažem se, trebalo bi. Ali ja bih radije da ta energija ide ka tome da kod bude toliko jasan i izražen da mu uopšte nisu potrebni komentari.

Netačni komentari su daleko gori od svih komentara. Oni zavaravaju i dovode u zabludu. Oni postavljaju očekivanja koja se nikada neće ispuniti. Oni govore o starim pravilima koja više nisu potrebna ili pravilima kojih ne treba da se držimo.

Istina se može naći samo na jednom mestu: u kodu. Samo kod vam zaista može reći šta se dešava. To je jedini izvor zaista tačnih informacija. Stoga, iako su komentari ponekad potrebni, potrošićemo značajnu energiju da ih minimalizujemo.

Komentari ne popravljaju loš kod

Jedan od najčešćih motiva za pisanje komentara je loš kod. Pišemo modul i znamo da je zbunjujući i neorganizovan. Znamo da je u neredu, pa kažemo sebi: „Bolje bi bilo da napišem komentare!“ Ne! Bolje je da kod učinite jasnim!

Jasan i izražajan kod s nekoliko komentara, daleko je bolji od pretrpanog i složenog koda s puno komentara. Umesto da trošite vreme na pisanje komentara koji objašnjavaju nered koji ste napravili, vreme upotrebite na čišćenje tog nereda.

Izražavajte se kodom

Sigurno postoje slučajevi kada je kod loše sredstvo za izražavanje. Nažalost, mnogi programeri su prihvatili da je kod retko, ako je uopšte, dobro sredstvo za izražavanje. Bez ikakve dileme to je pogrešan stav. Šta biste radije videli? Ovo:

```
// Proverava da li je zaposleni ispunio uslove za beneficije
if ((zaposleni.indikator & INDIKATOR _ SATI) &&
(zaposleni.imagodina > 65))
```

Ili ovo?

```
if (zaposleni.jesteIspunioUsloveZaBeneficije())
```

Potrebno vam je samo nekoliko sekundi razmišljanja da objasnite većinu vaših namera u kodu. U mnogim slučajevima to je jednostavno pisanje funkcije koja izražava istu stvar kao i komentar koji biste napisali.

Dobri komentari

Neki komentari su neophodni ili korisni. Pogledaćemo nekoliko stvari koje smatram vrednim svakog bita koji koriste. Međutim, imajte na umu da je jedini zaista dobar komentar onaj za koji ste našli način da ga ne napišete.

Pravni komentari

Ponekad nas korporativni standardi pisanja koda iz pravnih razloga prisiljavaju da pišemo određene komentare. Na primer, izjave o autorskim pravima i autorstvu su neophodne i razumne stvari koje treba staviti u komentar na početku svake izvorne datoteke.

Evo, na primer, standardnog zaglavlja komentara koje postavljamo na početak svake izvorne datoteke u FitNesseu. Srećan sam što mogu da kažem da naš IDE skriva ove komentare ne prikazujući ih.

```
// Copyright (C) 2003,2004,2005 by Object Mentor, Inc. All rights reserved.
// Released under the terms of the GNU General Public License version 2
// or later.
```

Komentari poput ovog ne bi trebalo da budu ugovori ili pravni dokumenti. Ako je moguće, pogledajte standardnu licencu ili druge postojeće dokumente, umesto da u komentar stavljate sve uslove i odredbe.

Informativni komentari

Ponekad je korisno u komentaru pružiti osnovne informacije. Primera radi, razmotrite ovaj komentar koji objašnjava povratnu vrednost apstraktne metode:

```
// Returns an instance of the Responder being tested.
protected abstract Responder responderInstance();
```

Komentar kao ovaj ponekad može biti koristan, ali bolje je da se, tamo gde je to moguće, za prenošenje informacija koristi ime funkcije. Na primer, u ovom slučaju komentar može biti suvišan preimenovanjem funkcije u `responderBeingTested`.

Evo slučaja koji je malo bolji:

```
// format matched kk:mm:ss EEE, MMM dd, yyyy
Pattern timeMatcher = Pattern.compile(
  "\\d*:\\d*:\\d* \\w*, \\w* \\d*, \\d*");
```

U ovom slučaju komentar nam daje do znanja da je regularni izraz namenjen da upari vreme i datum koji su formatirani funkcijom `SimpleDateFormat.format`, koristeći navedeni niz znakova. Ipak, moglo bi biti i bolje i jasnije, da je ovaj kod premešten u posebnu klasu koja pretvara formate datuma i vremena. Tada bi komentar verovatno bio suvišan.

Objašnjenje namere

Ponekad komentar nadilazi samo korisnu informaciju o primeni i daje dublji uvid. U sledećem slučaju vidimo zanimljivu odluku dokumentovanu komentarom. Upoređujući dva objekta, autor je odlučio da sortira predmete svoje klase veće od objekata bilo kojeg drugog.

```
public int compareTo(Object o)
{
  if(o instanceof WikiPagePath)
  {
    WikiPagePath p = (WikiPagePath) o;
    String compressedName = StringUtil.join(names, "");
    String compressedArgumentName = StringUtil.join(p.names, "");
    return compressedName.compareTo(compressedArgumentName);
  }
  return 1; // veći smo jer smo ispravnog tipa.
}
```

Evo još boljeg primera. Možda se ne slažete sa programerovim rešenjem problema, ali bar znate šta je on hteo da uradi.

```
public void testConcurrentAddWidgets() throws Exception {
  WidgetBuilder widgetBuilder =
    new WidgetBuilder(new Class[]{BoldWidget.class});
  String text = "'''bold text'''";
  ParentWidget parent =
    new BoldWidget(new MockWidgetRoot(), "'''bold text'''");
  AtomicBoolean failFlag = new AtomicBoolean();
  failFlag.set(false);

  //Ovo je najboji pokusaj da dobijemo bolje uslove
  //stvaranjem velikog broja niti (threads).
  for (int i = 0; i < 25000; i++) {
    WidgetBuilderThread widgetBuilderThread =
```

```
        new WidgetBuilderThread(widgetBuilder, text, parent, failFlag);
      Thread thread = new Thread(widgetBuilderThread);
      thread.start();
    }
    assertEquals(false, failFlag.get());
  }
```

Razjašnjenje

Ponekad je korisno pojasniti značenje nekog nejasnog argumenta ili vrednosti koja se vraća, u nešto što je čitljivo. Generalno, bolje je pronaći način da se taj argument ili povratna vrednost jasno pojave u sopstvenom obliku, ali kada je deo standardne biblioteke ili je u kodu koji ne možete da izmenite, komentar koji objašnjava može biti od koristi.

```
public void testCompareTo() throws Exception
{
  WikiPagePath a = PathParser.parse("PageA");
  WikiPagePath ab = PathParser.parse("PageA.PageB");
  WikiPagePath b = PathParser.parse("PageB");
  WikiPagePath aa = PathParser.parse("PageA.PageA");
  WikiPagePath bb = PathParser.parse("PageB.PageB");
  WikiPagePath ba = PathParser.parse("PageB.PageA");

  assertTrue(a.compareTo(a) == 0); // a == a
  assertTrue(a.compareTo(b) != 0); // a != b
  assertTrue(ab.compareTo(ab) == 0); // ab == ab
  assertTrue(a.compareTo(b) == -1); // a < b
  assertTrue(aa.compareTo(ab) == -1); // aa < ab
  assertTrue(ba.compareTo(bb) == -1); // ba < bb
  assertTrue(b.compareTo(a) == 1);  // b > a
  assertTrue(ab.compareTo(aa) == 1); // ab > aa
  assertTrue(bb.compareTo(ba) == 1); // bb > ba
}
```

Naravno, postoji značajan rizik da je komentar netačan. Prođite kroz prethodni primer i videćete koliko je teško da proverite da li su tačni. Ovo objašnjava zašto je razjašnjenje potrebno, ali i zašto je rizično. Pre nego što napišete komentare poput ovih, vodite računa o tome da budu tačni.

Upozorenje na posledice

Ponekad je korisno upozoriti druge programere na određene posledice. Primera radi, evo komentara koji objašnjava zašto je određeno testiranje isključeno:

```
// Ne pokreci osim ako nisi dokon
public void _ testWithReallyBigFile()
{
  writeLinesToFile(10000000);
```